**Available Online at www.ijcsmc.com**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**
**IMPACT FACTOR: 5.258**

***IJCSMC, Vol. 5, Issue. 5, May 2016, pg.765 – 770***

# Traffic Analysis in MapReduce

**Ms. Lavanya S R[1]**
[1]M.Tech Student, Department of Computer Science and Engineering,
New Horizon College of Engineering, Bangalore, Karnataka, India
[1] lavanya66kl@gmail.com

**Ms. Anjana Sharma[2]**
[2]Senior Assistant Professor, Department of Computer Science and Engineering,
New Horizon College of Engineering, Bangalore, Karnataka, India
[2] anjana.nhc@gmail.com

***Abstract--- MapReduce is a programming model, which can process the large set of data and produces the output. The MapReduce contains two functions to complete the work, those are Map function and Reduce function. The Map function will get assign fragmented data as input and then its emit intermediate data with key and send to this intermediate data with key to the Reducer, where Reducer will get the input from the Map function and then it compute the final output. In-between MapReduce the shuffle is there, which can be sort and shuffle the mapper inter mediate data and then it sends to the Reducer. In the Shuffle phase the large amount of traffic would occur and this can leads to even more traffic in the Reducer phase, because of this time and cost also increased, if one or two Mapper and Reducer is processing then it is ok, what if, the thousands of Map and Reducer are processing the job, no one has considered the traffic between the MapReduce, everyone is bothered only about load optimization and load balancing in Big Data, so in order to avoid the traffic in between the MapReduce in this we placed the Aggregator and Check function, in which checks the data come from the shuffle phase and then it removes the user irrelevant data and then reduces the data volume to process further and then sent to the Reducers, by placing the aggregator can reduces the traffic, cost and time and also it reduces the number of Reduce function to process the remaining task. This can be achieved by using the MapReduce distributed algorithm, and finally will show the performance analysis of the traffic reduction in the MapReduce.***

***Keywords: "MapReduce", "Shuffle", "Aggregator", "MapReduce Distributed", "Check function"***

## I. INTRODUCTION

Big Data refers to a large amount of data and in which the data is beyond the traditional data base software tool to capture, analyze and manage the data and also to store the data, in the limits of three dimensions are data volume, data variety and data velocity. Where Big Data is not only a raw data volume, which is considered by events, overall history and data transaction, the measuring of Big Data volume in terms of kilobyte, megabyte, gigabyte, terabyte, petabyte, exabyte zettabyte, yottabyte, to extract data volume from all these the Big Data analytics is very important. In the variety of Big Data contains a

 

structured, semi-structured and unstructured data, in which contains the different variety of data are Internet data in which contains social network, social medias and many others. Primary research data contains surveys, observations and experiences. And secondary research data contains consumer data, industry data reports, competitive and market place data, Business data, and location data contains a mobile device data. Geospatial data and Image data contains a video, satellite image, and supply chain data contains pricing, vendor catalogs, to store and process this variety of data is done by Big Data. To process this variety of data the velocity is very important, where in the Big Data can stores and process the data in the nano seconds.

Big Data Technology is most promising and it has been used Hadoop Distributed File System (HDFS) and MapReduce. The HDFS will provides a storage for clusters and once the data is stores in the HDFS then it breaks into number of small fragments and distributes those small pieces into number of servers which are present in the clusters, where each server stores a small fragment of complete data set and every fragment of data set can be replicated into more than one server, this replicated data set can be retrieved when the MapReduce is processing and in which one or more Map or Reducer fails to process.

A. MapReduce

The MapReduce is the programming model and also it is a framework which can process the large amount of dataset, which contains a two phases are map phase and reduce phase where map phase and reduce phase will get assign the work from the master, it is like a master worker problem, first master will get the data from the user to process and gives the output to the end users, so master will splits the input data into number of small fragments and then it will assign each fragments into a map function, where map and reduce phase contains a map function and reduce function, to process the work, once the master assigns each fragments into map function then map function will process and gives the intermediate values with key, to the shuffle phase where in which can shuffle and sort the intermediate values with key, which would present in between the map and reduce phase, once the shuffle phase is completed its work, then it sent to the reduce phase in this also master will take care of the assigning work to the reducer and then reducer will takes the intermediate data with key then it will process work and produces the final output to the user.

B. Existing System

MapReduce will process the large amount of data, and it can be process in the large set of clusters, and in between map and reduce phase shuffle phase is present, in which phase can sort and shuffle the large set of data and send to the reducer. During this MapReduce process finished within a second then also it produces traffic in the network while processing a large amount of data in MapReduce, due to increase of traffic in MapReduce will require more time than the actual time, and because of this, cost also increases to do the MapReduce job in the Big Data. Everyone consider only load optimization, load balancing in Big Data, and no one has the aware of traffic and cost in MapReduce phase.

C. Limitation of Existing System

- Even though Big Data technology is very advanced and it can have the MapReduce programming model to process the data, it has the traffic problem in it.
- MapReduce can process all the input data using map function and reduce function while doing this, it will not consider the traffic during many number of map nodes data are processing in the network at a time, and it is going through the same network switch at this point of time traffic would occur.
- In the existing system will not considered the data size with key to send from map node to the reduce node.
- Because of traffic problem in the MapReduce will increases the cost to process the data in the MapReduce.

## II. RELATED WORK

There are many research have been done regarding the optimization of the MapReduce job and the performance improvement of MapReduce job, load balancing while job processing.

Blanca et al.[18] author investigated and check the low network congestion even though high network utilization, by optimizing usage of network system may provides a better system performance, in this optimizing the network system and ignoring the data processing and combing of the data to reduce the traffic in the MapReduce. And Palanisamy et al [19] explained about the MapReduce resource allocation system and enhance the performance of MapReduce jobs by storing in to the cloud and mapped intermediate data to the local machines, and in this local awareness reduces the network traffic in the shuffle phase generated in the cloud data centre but it is generated large traffic between the MapReduce while processing the

 

task, because the large set of data has been shuffled once the map function completes its work during this produces the large amount of traffic while process this to the reducer.

Ibrahim et al. [20] is proposed the key partition, where in which checks the intermediate key partition and data distribution with key and value among all machines of map to reducer to checks the correctness of the data. And also the Liya et al. [21] have designed the algorithm of key distribution among the intermediate key value pair to improve the load balancing in the MapReduce. Lars et al. [22] is developed effective load balancing for MapReduce, but in this defined all above have been done a research, and by considering the load balancing in the MapReduce, but no one considering about traffic generated in the shuffle phase and the time required to process the MapReduce in the network containing the traffic in the shuffle phase and also, large amount of data is directly transfer to the reducer to gives the final output to the user.

Condie et al. [23] introduces the combiner function to combine the intermediate date with key value from the map and amount of data to be combined and sent to the reducer. Lin and Dyer [24] proposed an in-mapper combined scheme where in which mapper can preserves the state of intermediate data differently generated with key value pair until all input records have been processed in Map. These to researches have explained about the aggregation in the single Map, these are ignoring the aggregation of multiple map tasks.

## III. SYSTEM MODEL

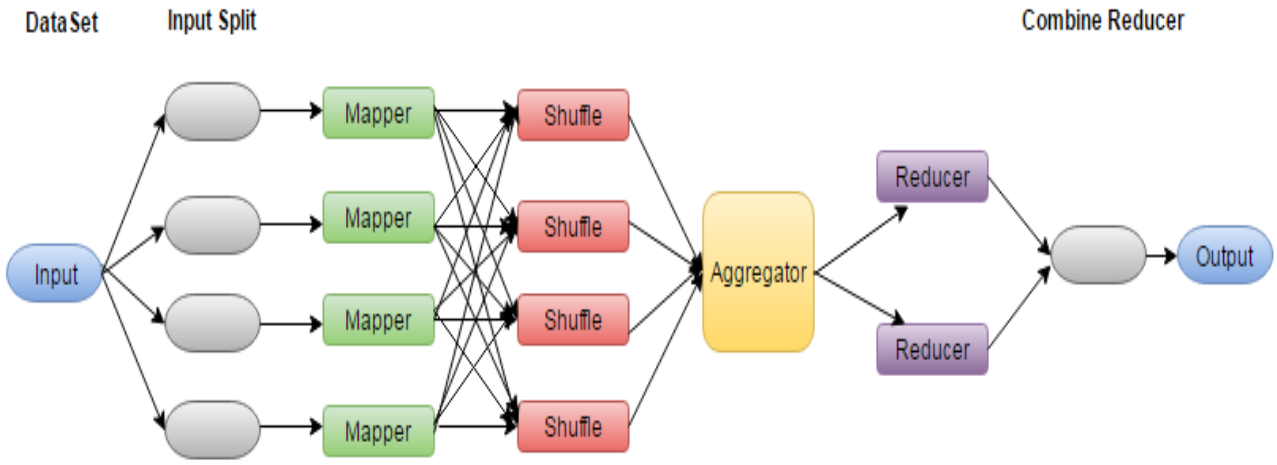


Fig.1 System Model of MapReduce

The user will upload dataset to the database to get the desired output from the MapReduce Programming model. In which first Map function will get assign the fragmented data and then it process the task and emits intermediate data with key, which can be sent to the Shuffle phase, where in which can shuffle and sort the data, then sent to the Aggregator, in which can reduce the data volume, traffic and time to process further then it sent to the Reducer, the reducer will compute the final output and the it stores in the database, in which user can retrieve their final output.

In the Fig.1 dataset is given as input and then it splits the number of input splits then it assigns each input splits into the Mapper, once the data is assigns to the mapper then it emits the intermediate data with random key and then it sends to the Shuffle in which phase data is shuffled and sorted, Aggregator will get the input from the Shuffle and then it removes the user irrelevant data and in this reduces the data volume and traffic then it sends to the Reducer. Where Reducer will compute the final output and then sent to user.

 

## IV. AGRREGATOR

In this section, gives the idea about the Aggregator and how it can do the tasks in between the MapReduce. And then how it reduces the traffic in between the MapReduce.

The Fig.2 explains about the Aggregator and Check function, that we are placed an Aggregator and Check function to reduce the traffic in the MapReduce. The Aggregator is to get input from the Shuffle phase and then it removes the user irrelevant data from the Check function by making check in Aggregator and then removing the data from Aggregator, and then sends the user relevant data to the Reducer. Once the Reducer phase gets the data from the Aggregator phase, then it computes the final output. By placing the Aggregator the number of Reducers can also decreased to process the further job in the MapReduce.

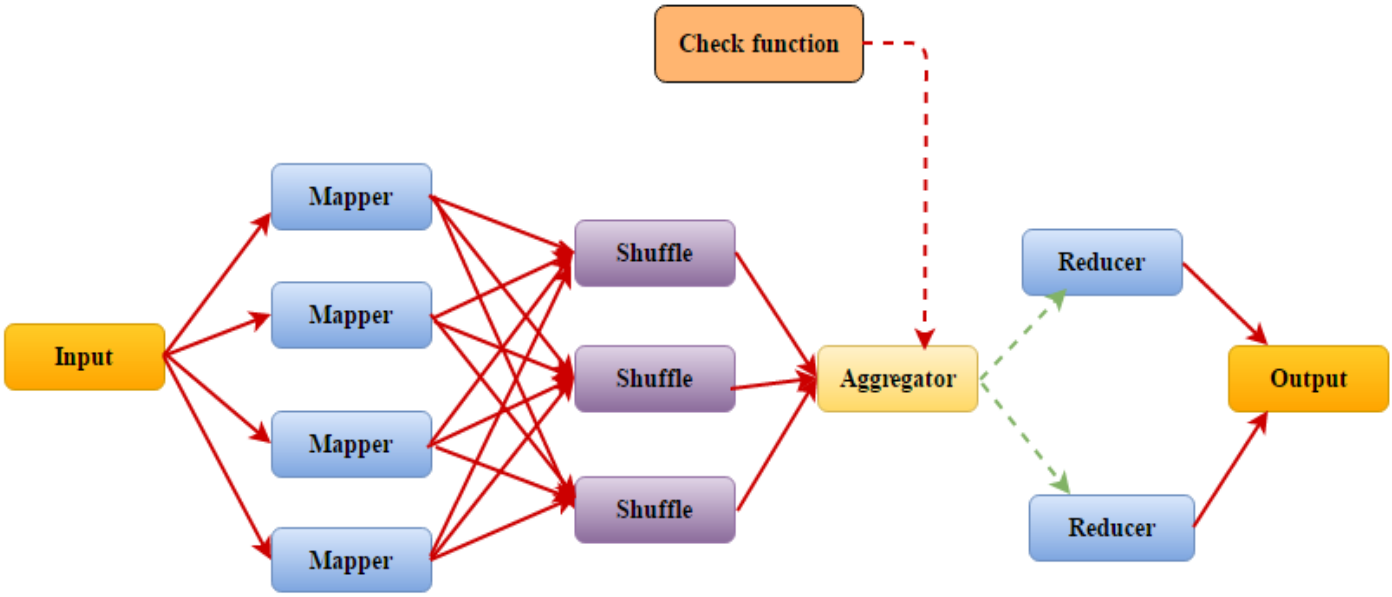


Fig.2 proposed Aggregator Check in MapReduce

**Distributed MapReduce Algorithm**

1. Set t=1 and variable v to the arbitrary positive value.
2. Set T to the random time.
3. For t < T do {
4. Distributively assign fragmented data to large number of clusters.
5. Map the data and emit intermediate data with key.
6. Shuffle and sort.
7. Calculate traffic, time and cost from Map to Shuffle.
   $Traffic(M\ to\ S) = \sum_{i=1}^{n} Mapper\ ij * DataVolume\ ij$
   Where M-> Mapper, S-> Shuffle, i->Mapper node and
   j-> Shuffle node
8. Place aggregator.
9. Remove irrelevant data and Calculate Traffic, Cost and time.
   $$Traffic(S\ to\ A) = \sum_{i=1}^{n} Shuffle\ \ jk * DataVolume\ jk$$
   Where S-> Shuffle, A-> Aggregator, j-> Shuffle node,

 

k-> Aggregator node

10. Reduce the data and Compute final output.
11. $Traffic(A\ to\ R) = \sum_{i=1}^{n} Aggregator\ \ kl * DataVolume\ kl$
    Where A-> Aggregator, R->Reducer, k-> Aggregator node,
    l-> Reducer node.

## V. RESULT

In this section explains about the performance in the MapReduce by placing the Aggregator in the MapReduce programming model. In the Fig.3 gives the idea about the traffic and time reduction in the MapReduce, in the Map to Shuffle phase is contains 56.66% of traffic, and it is reduced to 39.66% in the Aggregator to Reducer phase. Even in time also in the Shuffle phase is 34% is reduced to 28.72 % in the Reducer phase. So by placing the Aggregator in between the MapReduce, we achieve the Traffic, time and cost efficiency in the MapReduce programming model for the large set of data. In the Aggregator phase will removes the unwanted data and the sent to the Reducer by doing this can reduce the task of Reducers and cost also reduced to Process the large set of data in the MapReduce Programming Model.

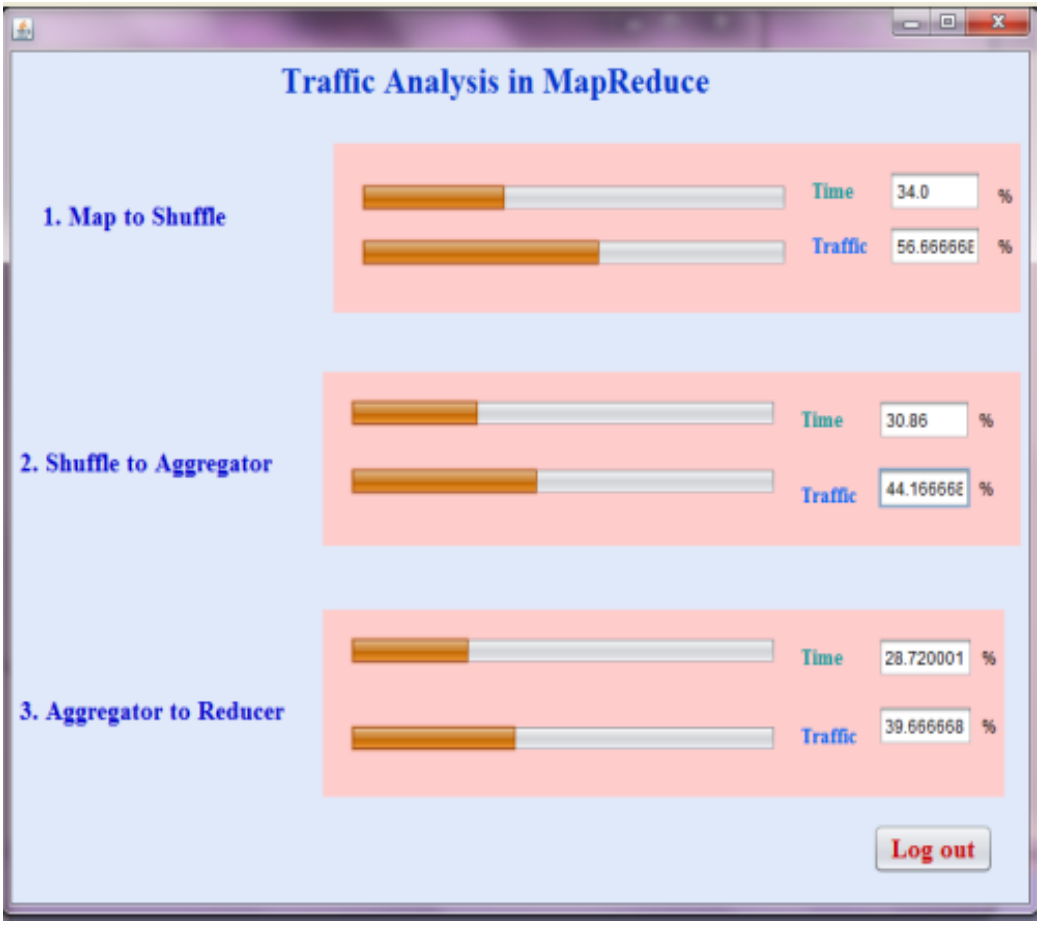


Fig.3 Traffic Analysis in MapReduce

## VI. CONCLUSIONS

MapReduce is a programming model where in which we are going to reduce the traffic, and time to process the large amount of data set in the MapReduce and will gives the traffic aware to the user and also will gives the time efficiency, this can be done by placing the aggregator in between the MapReduce and calculate reduced time, traffic in each step from Map phase to

 

Shuffle and Shuffle phase to Aggregator phase and then Aggregator to Reducer phase by using the MapReduce distributed algorithm. And also in this we will show the performance of traffic analysis in the Mapreduce by comparing the time and traffic in the Shuffle and Reducer phase by placing the Aggregator and using the MapReduce Distributed Algorithm.

## REFERENCES


[1]. J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.

[2]. W. Wang, K. Zhu, L. Ying, J. Tan, and L. Zhang, "Map task scheduling in mapreduce with data locality: Throughput and heavy-traffic optimality," in INFOCOM, 2013 Proceedings IEEE. IEEE, 2013, pp. 1609–1617.

[3]. F. Chen,M. Kodialam, and T. Lakshman, "Joint scheduling of processing and shuffle phases in mapreduce systems," in INFOCOM, **2012** Proceedings IEEE. IEEE, 2012, pp. 1143–1151.

[4]. Y. Wang, W. Wang, C. Ma, and D. Meng, "Zput: A speedy data uploading approach for the hadoop distributed file system," in Cluster Computing (CLUSTER), 2013 IEEE International Conference on. IEEE, 2013, pp. 1–5.

[5]. T. White, Hadoop: the definitive guide: the definitive guide. " O'Reilly Media, Inc.", 2009.

[6]. S. Chen and S. W. Schlosser, "Map-reduce meets wider varieties of applications," Intel Research Pittsburgh, Tech. Rep. IRP-TR-08-05, 2008.

[7]. J. Rosen, N. Polyzotis, V. Borkar, Y. Bu, M. J. Carey, M. Weimer, T. Condie, and R. Ramakrishnan, "Iterative mapreduce for large scale machine learning," arXiv preprint arXiv:1303.3517, 2013.

[8]. S. Venkataraman, E. Bodzsar, I. Roy, A. AuYoung, and R. S. Schreiber, "Presto: distributed machine learning and graph processing with sparse matrices," in Proceedings of the 8th ACM European Conference on Computer Systems. ACM, 2013, pp. 197–210.

[9]. Matsunaga, M. Tsugawa, and J. Fortes, "Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications," in eScience, 2008. eScience'08. IEEE Fourth International Conference on. IEEE, 2008, pp. 222–229.

[10]. J. Wang, D. Crawl, I. Altintas, K. Tzoumas, and V. Markl, "Comparison of distributed data-parallelization patterns for big data analysis: A bioinformatics case study," in Proceedings of the Fourth International Workshop on Data Intensive Computing in the Clouds (DataCloud), 2013.

[11]. R. Liao, Y. Zhang, J. Guan, and S. Zhou, "Cloudnmf: A mapreduce implementation of nonnegative matrix factorization for largescale biological datasets," Genomics, proteomics & bioinformatics, vol. 12, no. 1, pp. 48–51, 2014.

[12]. G. Mackey, S. Sehrish, J. Bent, J. Lopez, S. Habib, and J. Wang, "Introducing map-reduce to high end computing," in Petascale Data Storage Workshop, 2008. PDSW'08. 3rd. IEEE, 2008, pp. 1–6.

[13]. W. Yu, G. Xu, Z. Chen, and P. Moulema, "A cloud computing based architecture for cyber security situation awareness," in Communications and Network Security (CNS), 2013 IEEE Conference on. IEEE, 2013, pp. 488–492.

[14]. J. Zhang, H. Zhou, R. Chen, X. Fan, Z. Guo, H. Lin, J. Y. Li, W. Lin, J. Zhou, and L. Zhou, "Optimizing data shuffling in dataparallel computation by understanding user-defined functions," in Proceedings of the 7th Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, USA, 2012.

[15]. F. Ahmad, S. Lee,M. Thottethodi, and T. Vijaykumar, "Mapreduce with communication overlap," pp. 608–620, 2013.

[16]. H.-c. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker, "Mapreduce-merge: simplified relational data processing on large clusters," in Proceedings of the 2007 ACM SIGMOD international con- ference on Management of data. ACM, 2007, pp. 1029–1040.

[17]. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, J. Gerth, J. Talbot, K. Elmeleegy, and R. Sears, "Online aggregation and continuous query support in mapreduce," in Proceedings of the 2010 ACM SIGMOD International Conference on Management of data. ACM, 2010, pp. 1115–1118.

[18]. A. Blanca and S. W. Shin, "Optimizing network usage in mapreduce scheduling."

[19]. B. Palanisamy, A. Singh, L. Liu, and B. Jain, "Purlieus: localityaware resource allocation for mapreduce in a cloud," in Proceed- ings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011, p. 58.

[20]. S. Ibrahim, H. Jin, L. Lu, S. Wu, B. He, and L. Qi, "Leen: Locality/fairness-aware key partitioning for mapreduce in the cloud," in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. IEEE, 2010, pp. 17–24.

[21]. L. Fan, B. Gao, X. Sun, F. Zhang, and Z. Liu, "Improving the load balance of mapreduce operations based on the key distribution of pairs," arXiv preprint arXiv:1401.0355, 2014.

[22]. S.-C. Hsueh, M.-Y. Lin, and Y.-C. Chiu, "A load-balanced mapreduce algorithm for blocking-based entity-resolution with multiple keys," Parallel and Distributed Computing 2014, p. 3, 2014.

[23]. T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce Online." In NSDI, vol. 10, no. 4, p. 20. 2010.


 